Système à base de connaissances pour quelques problèmes aux valeurs propres

Jean-François Carpraux

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Système à base de connaissances pour quelques problèmes aux valeurs propres

Jean-François Carpraux

N° 1975

Avril 1993

PROGRAMME 6

Calcul scientifique,
modélisation
et logiciels numériques

ISSN 0249-6399

INRIA
RENNES

Système à base de connaissances pour quelques problèmes aux valeurs propres

Jean-François Carpraux

Programme 6 — Calcul scientifique, modélisation et logiciel numérique
Projet Aladin

Rapport de recherche n ° 1975 — Avril 1993 — 38 pages

Résumé : Ce rapport présente un système à base de connaissances pour la librairie LAPACK [1] d'aide au choix de procédure et à la validation de résultat. Le système s'intéresse aussi à la qualité numérique du résultat. Il peut à la fois conseiller une procédure et vérifier la validité d'une procédure donnée pour résoudre un problème de l'usager. Le système repose sur le logiciel SHIRKA [22] écrit en Le-Lisp qui met en œuvre des bases de connaissances à objets avec classification. Toute la connaissance est contenue dans des objets organisés en hiérarchies.
Mots-clé : Valeur propre, Conditionnement, Système Expert, Objets.

(Abstract: pto)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 38 38 32

A Knowledge-Based System for Eigenvalue Problems

Abstract: This paper presents a knowledge-based system devoted to the LAPACK library [1] to help the user in selecting a numerical procedure and in validating a result. Our system is also concerned with the numerical quality of the result. It can either select a routine or check the validity of a given routine to solve a user's problem. Our system is based on the development shell SHIRKA [22], written in Lisp, which provides means to describe knowledge bases and to apply a classification mechanism upon them. All the knowledge is contained into the objects which are organized into hierarchies.
Key-words: Eigenvalue, Condition number, Expert System, Objects.

Remerciements :

Je tiens à remercier Jocelyne Erhel, Chercheur à l'IRISA, et François Rechenmann, Directeur de Recherche à l'IRIMAG, pour l'aide et les conseils qu'ils ont apportés dans l'élaboration de ce travail.

Table des matières

Introduction

Les bibliothèques numériques ont atteint un degré de complexité tel qu'il est difficile pour l'usager de sélectionner la ou les procédures les mieux adaptées au problème qu'il veut résoudre. Aussi faut-il fournir avec une bibliothèque un mode d'emploi intelligent qui guide l'usager. Par exemple la société NAG développe des systèmes experts d'aide au choix de procédure qui couvrent leur bibliothèque scientifique [10, 13, 15]. Une expérience conduite au CNES [23] a conduit à la réalisation d'un prototype pour la résolution de systèmes linéaires. Il est également possible de générer les séquences d'appel aux procédures sélectionnées comme par exemple dans le système d'aide à la bibliothèque MODULEF [18].

A l'inverse, il se peut que l'usager ait utilisé une procédure de bibliothèque et souhaite vérifier que celle-ci résout effectivement le problème posé, par exemple que les données sont bien dans le domaine de validité de la méthode numérique choisie. Un système d'aide devra donc fonctionner en quelque sorte à l'envers, c'est-à-dire valider a posteriori le choix d'une procédure en fonction du problème à résoudre.

La précision d'un résultat de calcul scientifique dépend de l'erreur sur les données, du conditionnement du problème et de la stabilité numérique de l'algorithme de résolution. Quelques outils logiciels permettent maintenant d'évaluer l'erreur sur le résultat. Par exemple, le système d'aide [17] choisit et exécute des procédures d'encadrement qui ont recours à l'arithmétique d'intervalles pour fournir un encadrement sûr du résultat. L'atelier de qualité numérique AQUARELS [8] a pour objectif de rassembler dans une structure cohérente et conviviale diverses solutions pour contrôler ou améliorer la précision des calculs. Un système d'aide pourrait alors suggérer l'utilisation de tel ou tel outil pour évaluer les diverses composantes de l'erreur finale.

A l'IRISA, nous avons développé un prototype appelé SESAME (Système Expert pour la Sélection et la vAlidation de MÉthodes numériques) qui vise à remplir trois fonctions :

- sélection de procédure,
- vérification de procédure,
- validation de résultat.

Nous avons choisi la bibliothèque LAPACK [1] comme cible numérique et plus particulièrement un sous-ensemble de LAPACK dédié au calcul de valeurs et vecteurs propres qui contient une centaine de procédures. A notre avis, ce domaine est suffisamment vaste et complexe pour montrer la faisabilité de notre approche.

Le choix de la bibliothèque LAPACK permet de simplifier et de conclure la validation de résultat. En effet, il n'est pas besoin ici de se préoccuper des erreurs d'arrondi puisque les algorithmes codés dans LAPACK sont stables. De plus, la bibliothèque fournit des procédures pour estimer le conditionnement des valeurs ou vecteurs propres calculés. Il est donc possible d'estimer par une formule approximative l'erreur sur le résultat dès que l'on connaît l'erreur sur la matrice de départ.

Le système est basé sur des techniques d'intelligence artificielle. En effet, un système expert présente l'avantage de pouvoir expliquer le cheminement suivi pour aboutir à la conclusion proposée. Il comporte en outre une interface avec l'utilisateur qui en facilite la manipulation. De plus, la modélisation des connaissances en est simplifiée et peut évoluer relativement facilement.

L'expertise numérique est modélisée par une base de connaissances contenant des objets qui représentent les entités mathématiques à manipuler. Ces objets sont organisés en hiérarchies qui peuvent être assez complexes grâce à l'héritage multiple. Le moteur d'inférence est le mécanisme de classification qui s'avère très efficace dans le contexte de calcul scientifique. Lors de la classification, des valeurs d'attributs peuvent être inférées grâce aux valeurs par défaut ou aux attachements procéduraux. Ainsi, la sélection d'une procédure fonctionne un peu à la manière d'un arbre de décision. Il suffit d'inférer dans la feuille classée sûre le nom de la procédure à utiliser. Mais la même base de connaissances permet également de vérifier une procédure, grâce à la notion de classe d'attributs. Cette fois, la classification prend en compte le nom de la procédure et il faut qu'une feuille soit classée sûre pour que la procédure de l'usager soit valide.

Le système repose sur le logiciel SHIRKA [22] écrit en Le-Lisp qui met en œuvre des bases de connaissances à objets avec classification.

La première partie décrit la problématique numérique et la formalise pour aboutir à une modélisation par objets. La seconde partie résume les fonctionnalités de SHIRKA qui ont été exploitées pour développer le système, puis décrit celui-ci et la façon de l'utiliser.

Première partie
Expertise numérique

On s'intéresse au problème suivant :

Etant donnée $A \in \mathbb{C}^{n \times n}$, une matrice carrée d'ordre n, trouver des éléments $\lambda \in \mathbb{C}$ et/ou $x \in \mathbb{C}^n$ tels que :
λ soit une valeur propre de A (i.e. $\det(A - \lambda I) = 0$)
et x soit un vecteur propre de A associé à λ (i.e. $Ax = \lambda x$).

On se restreint au cas des matrices traitées dans Lapack, c'est-à-dire des matrices d'ordre n "petit" (moins de 5000), et qui sont des matrices bandes, unidimensionnelles (stockées dans un tableau) ou denses. Le cas des matrices creuses n'est pas traité dans Lapack.

De plus, on ne traite pas le problème aux valeurs propres généralisé, celui-ci pouvant bien entendu être une extension du système expert.

On se propose de créer un système d'aide proposant les choix suivants :
- Déterminer la (les) procédure(s) Lapack à utiliser pour résoudre un problème aux valeurs propres donné.
- Vérifier qu'une procédure fournie par l'usager résout effectivement le problème posé.
- Valider le résultat d'une procédure.

1 Etude du problème

Le but de cette partie est de déterminer les différentes méthodes de résolution de ce problème et les procédures Lapack correspondantes, en fonction de certaines propriétés des données. Dans ce qui suit, nous indiquons le nom des procédures en remplaçant la première lettre par "x" car celle-ci dépend uniquement du type de calcul effectué (simple ou double précision) et du type de la matrice (réelle ou complexe). En effet, c'est un S dans le cas réel simple précision, un D dans le cas réel double précision, un C dans le cas complexe simple précision et enfin un Z dans le cas complexe double précision.

1.1 Méthodes de calcul des valeurs propres

La première étape est de trouver une matrice plus simple qui aura les mêmes valeurs propres que A. Pour cela on applique la méthode de Householder [6, 27], qui construit une matrice S unitaire ($S^* = S^{-1}$) telle que $H = S^*AS$ soit une matrice de **Hessenberg supérieure**, c'est à dire une matrice qui n'admet que des zéros en dessous de sa sous-diagonale. Cette factorisation conserve la symétrie. Ainsi, pour A symétrique (par abus de langage on dit qu'une matrice hermitienne est une matrice symétrique), la matrice transformée est **tridiagonale symétrique**. Dans le cas hermitien, on peut très facilement se ramener à une matrice **tridiagonale réelle symétrique**.
Dans le cas non symétrique, la procédure Lapack réalisant cette transformation est xGEHRD, tandis que dans le cas réel symétrique (respectivement hermitien), c'est xSYTRD (resp. xHETRD) pour une matrice dense, xSPTRD (resp. xHPTRD) pour une matrice tableau et xSBTRD (resp. xHBTRD) pour une matrice bande.

On vient de construire une matrice $H = S^*AS$ plus simple que A et qui a les mêmes valeurs propres que celle-ci.

Il nous suffit, désormais, de considérer les 2 types de matrices suivants :

1. Hessenberg supérieure
2. Tridiagonale réelle symétrique

Il n'existe pas de méthode simple pour calculer une partie du spectre d'une matrice de Hessenberg (réelle ou complexe), par conséquent, on calcule toujours toutes les valeurs propres. On utilise la **méthode du double QR avec translations** [27] (procédure xHSEQR). Cette méthode est une méthode itérative qui consiste à construire une suite de matrices H_k unitairement semblables (c'est-à-dire que pour tout k il existe une matrice Q_k unitaire telle que $H_k = Q_k^* H Q_k$). Cette suite de matrices H_k converge vers une matrice triangulaire supérieure par blocs appelée forme de Schur de la matrice H. Chaque sous-matrice diagonale correspond aux valeurs propres d'un même module. Les translations permettent d'accélerer cette convergence vers la forme de Schur.
En pratique, les blocs diagonaux sont soit de dimension 1 (valeur propre

simple), soit de dimension 2 (valeurs propres complexes conjuguées).

Dans le cas d'une matrice tridiagonale réelle symétrique, il est possible de calculer une valeur propre à la fois par la **méthode de Bissection** (procédure xSTEBZ), qui permet d'approcher d'aussi près qu'on veut cette valeur propre, ou de toutes les calculer par la **méthode QL** (procédure xSTEQR si l'on désire calculer des vecteurs propres, et procédure xSTERF qui est plus rapide que la précédente si l'on désire seulement calculer les valeurs propres). Cette méthode n'est autre que la méthode QR où L est triangulaire inférieure. On la préfère à la méthode QR car elle présente quelques petits avantages du point de vue de la programmation.
Heuristiquement, on a remarqué qu'il était moins coûteux d'utiliser la méthode QL dès qu'on voulait calculer plus de 25% des valeurs propres.

Lorsqu'on utilise la méthode QR (ou QL), on construit une matrice Q unitaire ($Q^* = Q^{-1}$), telle que $T = Q^*HQ$ soit sous forme de Schur.

Remarque: La forme de Schur d'une matrice symétrique est une matrice diagonale.

1.2 Méthodes de calcul des vecteurs propres

Après avoir calculé les valeurs propres de la matrice A par la méthode QR (ou QL), on peut en calculer tous les vecteurs propres par la **méthode de substitution retour** (procédure xTREVC dans le cas non symétrique et xSTEQR dans le cas symétrique) dont le principe est le suivant :
soit T la matrice sous forme de Schur construite par la méthode QR, on en construit une base de vecteurs propres (facile et peu coûteux) et on récupère ceux de A en les multipliant par SQ, où S et Q ont été définies précédemment. Il faudra donc avoir stocké ces matrices au préalable.

Si l'on s'intéresse uniquement au calcul d'un vecteur propre associé à une valeur propre dont on a calculé une bonne approximation λ', ou si les valeurs propres n'ont pas été calculées par la méthode QR (ou QL), on utilise la **méthode des itérations inverses** (procédure xSTEIN dans le cas symétrique et xHSEIN dans le cas non symétrique). Cette méthode itérative est définie de la façon suivante :

Soit u_0 un vecteur arbitraire non nul

Pour $k \geq 0$, on définit u_{k+1} par $(T - \lambda' I)u_{k+1} = u_k$

On montre que u_k converge vers le vecteur propre associé à la valeur propre la plus proche de λ', sauf si u_0 est orthogonal à ce vecteur propre.

De façon heuristique, si on désire calculer moins de 25% des vecteurs propres, on décide d'utiliser la méthode des itérations inverses .

1.3 Formalisation

On vient de voir que la méthode à utiliser dépend de certaines propriétés de la matrice (symétrique ou non, réelle ou complexe...) et du nombre de valeurs et vecteurs propres que l'on désire calculer.

Récapitulons les différentes procédures conseillées en fonctions des propriétés du problème initial :

1. **Transformation de la matrice** :
   (a) Matrice non symétrique : xGEHRD.
   (b) Matrice symétrique :
      i. Dense : xSYTRD (réel), xHETRD (complexe).
      ii. Tableau : xSPTRD (réel), xHPTRD (complexe).
      iii. Bande : xSBTRD (réel), xHBTRD (complexe).
2. **Calcul des valeurs propres** :
   (a) Matrice de Hessenberg : xHSEQR.
   (b) Matrice tridiagonale réelle symétrique
      i. xSTEBZ pour calculer moins de 25% des valeurs propres.
      ii. Calcul de plus de 25% des valeurs propres :
         A. xSTEQR si plus de 25% des vecteurs propres sont désirés.
         B. xSTERF sinon.
3. **Calcul de moins de 25% des vecteurs propres** :
   (a) Matrice de Hessenberg : xHSEIN.

(b) Matrice tridiagonale réelle symétrique : xSTEIN.

4. **Calcul de plus de 25% des vecteurs propres :**

   (a) Matrice de Hessenberg : xTREVC.

   (b) Matrice tridiagonale réelle symétrique

      i. xSTEIN si les valeurs propres ont été calculées par xSTEBZ.

      ii. xSTEQR si les valeurs propres ont été calculées par elle même.

1.4 Les driver et expert routines

Il existe, dans Lapack, des procédures appelées driver et expert routines qui permettent le calcul de valeurs propres et éventuellement des vecteurs propres associés pour des problèmes standard. Pour certaines de ces procédures, il est également possible d'estimer les conditionnements des valeurs et vecteurs propres calculés.

Voici la liste de ces procédures et ce qu'elles font :

1. **Cas symétrique**

   (a) Calcul de toutes les valeurs propres et éventuellement de tous les vecteurs propres

      i. Matrice dense : xSYEV (réel), xHEEV (complexe).

      ii. Matrice tableau : xSPEV (réel), xHPEV (complexe).

      iii. Matrice bande : xSBEV (réel), xHBEV (complexe).

      iv. Matrice tridiagonale réelle : xSTEV.

   (b) Calcul de quelques valeurs propres et éventuellement des vecteurs propres associés.

      i. Matrice dense : xSYEVX (réel), xHEEVX (complexe).

      ii. Matrice tableau : xSPEVX (réel), xHPEVX (complexe).

      iii. Matrice bande : xSBEVX (réel), xHBEVX (complexe).

      iv. Matrice tridiagonale réelle : xSTEVX.

2. **Cas non symétrique**

   (a) Calcul de toutes les valeurs propres et éventuellement de tous les vecteurs propres :

i. Sans calcul des conditionnements : xGEEV.

ii. Avec calcul des conditionnements : xGEEVX.

(b) Calcul de toutes les valeurs propres et éventuellement de tous les vecteurs de Schur :

i. Sans calcul des conditionnements : xGEES.

ii. Avec calcul des conditionnements : xGEESX.

Ces bases de connaissances servent à déterminer des procédures, mais elles ne permettent pas la validation de celles-ci. En effet, ces bases ne donnent que la procédure la plus adaptée à un problème donné, elles ne donnent pas la liste des procédures valables. Or, on ne peut pas interdire à l'usager d'utiliser une procédure pour la seule raison que celle-ci n'est pas la "meilleure". On va donc, maintenant, donner deux bases qui déterminent une liste de procédures utilisables pour un problème donné.

1.5 Validation de procédure

Il faut remarquer que la plupart de ces procédures ont une implémentation qui dépend de la structure de stockage de la matrice (matrice symétrique, matrice bande, matrice tableau, ...).
On interdira donc, par exemple, d'utiliser une procédure traitant d'une matrice non symétrique si la matrice considérée est symétrique, bien que celle-ci soit valable mathématiquement.

Par contre, il faudra prendre en compte le fait qu'on ne peut pas interdire à un usager d'utiliser une procédure qui calcule toutes les valeurs propres (ou tous les vecteurs propres), même s'il n'en veut qu'une partie, ou de calculer tous les éléments propres un par un ...

Faisons la liste des procédures valables **(autres que la plus adaptée)** pour un problème donné :

1. **Calcul de valeurs propres d'une matrice symétrique** :

   (a) Toutes : xSTEBZ.

   (b) Quelques unes : xSTERF, xSTEQR.

2. **Calcul des vecteurs propres** :

(a) Cas symétrique

i. Tous : xSTEIN.

ii. Quelques uns : xSTEQR (si les valeurs propres ont été calculées par cette même procédure).

(b) Cas non symétrique

i. Tous : xHSEIN.

ii. Quelques uns : xTREVC si la matrice est sous forme de Schur.

La liste de toutes les driver et expert routines utilisables pour un problème donné est :

1. **Calcul de toutes les valeurs propres (et vecteurs propres) ou seulement de quelques-unes (et les vecteurs propres correspondant) dans le cas où la matrice est symétrique**

   (a) Matrice dense réelle : xSYEV, xSYEVX.

   (b) Matrice dense complexe : xHEEV, xHEEVX.

   (c) Matrice tableau réelle : xSPEV, xSPEVX.

   (d) Matrice tableau complexe : xHPEV, xHPEVX.

   (e) Matrice bande réelle : xSBEV, xSBEVX.

   (f) Matrice bande complexe : xHBEV, xHBEVX.

   (g) Matrice tridiagonale réelle : xSTEV, xSTEVX.

2. **Cas non symétrique :** seule la procédure la meilleure est utilisable.

2 Validation de résultat

Le conditionnement mesure la variation du résultat en fonction d'une variation sur les données du problème.
On suppose, ici, que les calculs effectués sont exacts. Ainsi, les résultats calculés correspondent exactement aux données.
On dit qu'un problème est bien conditionné lorsqu'il n'est pas trop sensible aux perturbations sur les données initiales, de même, il est dit mal conditionné lorsqu'il est très sensible à ces perturbations.

Le conditionnement d'un problème aux valeurs propres consiste à mesurer la variation des valeurs propres et vecteurs propres en fonction d'une variation ΔA sur la matrice A quelconque. Ceci est étudié de façon détaillée dans [4]. On rappelle ici les résultats utilisés dans Lapack et dans le système expert.

2.1 Conditionnement dans le cas symétrique

Les valeurs propres, qu'elles soient simples ou multiples, sont bien conditionnées avec un conditionnement égal à 1. Le conditionnement des vecteurs propres ou sous-espaces invariants ne dépend que de la distance du bloc de valeurs propres au reste du spectre.

$$cond\,(\lambda) = 1$$

$$cond\,(x) = dist\,(\lambda,\, sp(A) - \{\lambda\})$$

$$cond\,(M) = dist\,(\sigma,\, sp(A) - \{\sigma\})$$

où λ est une valeur propre quelconque, x un vecteur propre associé à la valeur propre simple λ, M un sous-espace invariant associé au bloc de valeurs propres σ.

2.2 Conditionnement dans le cas non symétrique

Les formulations des conditionnements sont complexes [4] et sont liées à la forme de Jordan des matrices qui est très difficile à calculer en pratique [12]. C'est pourquoi, on présente, ici, un moyen de trouver une estimation des conditionnements qui est beaucoup moins coûteux que de les calculer exactement.
La méthode proposée [2] s'applique sur des matrices non symétriques mises sous forme de Schur (matrice triangulaire supérieure par blocs fournie par la méthode QR, procédure xHSEQR).

Soit T (matrice $n \times n$) sous forme de Schur :

$$T = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$$

où T_{11} (matrice $m \times m$) est associée à m valeurs propres de même module dont on veut calculer le conditionnement.

Dans le cas de valeurs propres simples conjuguées complexes la matrice T_{11} sera 2×2.
En pratique, pour une valeur propre de multiplicité m, T_{11} aura m valeurs propres distinctes voisines (erreurs d'arrondi), c'est pourquoi, on parlera de bloc de valeurs propres plutôt que de valeur propre multiple.

Soient :

- ΔT une perturbation de la matrice T,
- $\epsilon_2 = \|\Delta T\|_2$ et $\epsilon_F = \|\Delta T\|_F$

2.2.1 Valeur propre simple et vecteur propre associé

Soient :

- λ une valeur propre de T,
- λ' la valeur propre de $T + \Delta T$ correspondant à λ.

On peut montrer [27] :

$$|\lambda - \lambda'| \leq \epsilon_2 \, \|P\|_2 + O(\epsilon_2^2)$$

$$\text{où } \|P\|_2 = \frac{|x^* \, y|}{\|x\|_2 \, \|y\|_2}$$

où x et y sont respectivement les vecteurs propres à droite et à gauche de T associés à λ ($Tx = \lambda x, \;\; T^* y = \overline{\lambda} y$).

Dans ce cas, le calcul de $\|P\|_2$ se fait simplement grâce aux vecteurs propres à droite et à gauche associés à la valeur propre considérée.

Soient x le vecteur propre à droite de T associé à λ et x' le vecteur propre perturbé correspondant, on montre [7]

$$\theta_{max}(x, x') \leq \frac{2 \, \epsilon_F}{sep \, (T_{11}, T_{22})} + O(\epsilon_F^2)$$

où $sep \, (T_{11}, T_{22})$ est défini de la manière suivante :

Si T_{11} est 1×1 (valeur propre réelle), on a $T_{11} = \lambda$ et alors [25] :

$$sep\,(T_{11}, T_{22}) = \min_{\|x\|_2=1} \|(\lambda\, I - T_{22})\, x\|_2$$

Si T_{11} est un bloc 2×2 (valeur propre complexe), on triangularise ce bloc en utilisant une rotation unitaire et on obtient :

$$T' = \begin{pmatrix} \lambda & t_{12} \\ 0 & T'_{22} \end{pmatrix}$$

$$\text{on a alors} \qquad sep\,(T_{11}, T_{22}) = \min_{\|x\|_2=1} \|(\lambda\, I - T'_{22})\, x\|_2$$

Dans les deux cas,

$$sep\,(T_{11}, T_{22}) = \min_{\|x\|_2=1} \|(\lambda\, I - T'')\, x\|_2$$

où $T'' = T_{22}$ si la valeur propre est réelle et $T'' = T'_{22}$ si la valeur propre est complexe.

On peut donner une estimation de $sep\,(T_{11}, T_{22})$ en calculant $\|K^{-1}\|_1$ où $K = (T'' - \lambda\, I)$ puisque $sep\,(T_{11}, T_{22}) = \|K^{-1}\|_2^{-1}$ et

$$\frac{1}{\sqrt{n-1}}\, \|K^{-1}\|_1 \;\leq\; \|K^{-1}\|_2 \;\leq\; \sqrt{n-1}\, \|K^{-1}\|_1$$

2.2.2 Bloc de valeurs propres et sous-espace invariant associé

Le conditionnement d'une valeur propre de multiplicité $m > 1$ fait intervenir l'indice de celle-ci [4] (taille du plus grand bloc de Jordan qui lui est associé). Or, l'indice est très difficile à calculer en pratique. C'est pourquoi, on considère les quantités suivantes :

- $\overline{\lambda} = \frac{trace(T_{11})}{m}$ la moyenne du bloc des valeurs propres de T_{11},
- $\overline{\lambda}'$ la moyenne du bloc des valeurs propres de $T + \Delta T$ correspondant à $\overline{\lambda}$.

et on montre le résultat suivant [16] :

$$|\overline{\lambda} - \overline{\lambda}'| \leq \epsilon_2\, \|P\|_2 + O(\epsilon_2^2)$$

où le projecteur spectral P est défini par :

$$P = \begin{pmatrix} I_m & R \\ 0 & 0 \end{pmatrix}$$

où R est solution de l'équation de Sylvester $T_{11}R - RT_{22} = T_{12}$.

On a $\|P\|_2 = \sqrt{1 + \|R\|_2^2}$, dont le calcul est assez coûteux en pratique, c'est pourquoi on calcule plutôt $\|P\|' = \sqrt{1 + \|R\|_F^2}$ puisque :

$$\|R\|_2^2 \leq \|R\|_F^2 \leq n \, \|R\|_2^2$$

$$\text{donc } \frac{1}{\sqrt{1 + \|R\|_F^2}} \leq \frac{1}{\|P\|_2} \leq \frac{\sqrt{n}}{\sqrt{1 + \|R\|_F^2}}$$

Soient :

- M le sous-espace invariant à droite de T_{11} et M' le sous-espace perturbé correspondant,
- $\theta_{max}(M, M')$ l'angle entre les sous-espaces M et M' défini par :

$$\theta_{max}(X, Y) = \max_{x \in X, x \neq 0} \min_{y \in Y, y \neq 0} \theta(x, y) = \max_{y \in Y, y \neq 0} \min_{x \in X, x \neq 0} \theta(x, y)$$

On montre [7]

$$\theta_{max}(M, M') \leq \frac{2 \, \epsilon_F}{sep \, (T_{11}, T_{22})} + O(\epsilon_F^2)$$

où $sep \, (T_{11}, T_{22})$ est défini par :

$$sep \, (T_{11}, T_{22}) = \min_{X \neq 0} \frac{\|T_{11}X - XT_{22}\|_F}{\|X\|_F} = \sigma_{min}(K)$$

$$\text{où } K = I_{n-m} \otimes T_{11} - T_{22}^T \otimes I_m$$

où σ_{min} désigne la plus petite valeur singulière et où $A \otimes B$ désigne le produit tensoriel (ou de Kronecker) des matrices A et B.

Une estimation de $sep\,(T_{11}, T_{22})$, est donnée par $\|K^{-1}\|_1^{-1}$: en effet,

$$sep\,(T_{11}, T_{22}) \;=\; \sigma_{min}(K) \;=\; \|K^{-1}\|_2^{-1}$$

$$\text{et } \frac{1}{\sqrt{m\,(n-m)}}\,\|K^{-1}\|_1 \;\leq\; \|K^{-1}\|_2 \;\leq\; \sqrt{m\,(n-m)}\,\|K^{-1}\|_1$$

2.3 Estimation des conditionnements

Soit T (matrice $n \times n$ non symétrique) sous forme de Schur :

$$T = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$$

où les valeurs propres de T_{11} (matrice $m \times m$) sont celles qu'on a décidé de regrouper. Par exemple, on regroupe les valeurs propres séparées par une distance de l'ordre de la précision de l'ordinateur.

Etant donné les majorations précédentes, il nous suffit de calculer les quantités $\|P\|_2$ et $sep\,(T_{11}, T_{22})$ pour estimer respectivement les conditionnements des valeurs et des vecteurs propres.
On préfère estimer les inverses des conditionnements ($\|P\|_2^{-1}$ et $sep\,(T_{11}, T_{22})$), puisque ces deux quantités restent bornées et sont comprises entre 0 et 1. Les valeurs proches de 0 correspondent à des conditionnements très grands.

L'estimation du conditionnement d'une valeur propre simple coûte $O(n)$ opérations, et de l'ordre de $O(n^2)$ opérations pour le vecteur propre associé. La procédure Lapack correspondant à ces estimations est xTRSNA. Grâce à un paramètre de cette procédure, on peut choisir de n'estimer qu'un des deux conditionnements (valeur et vecteur propre) ou les deux.

Le calcul du conditionnement d'un bloc de valeurs propres ou du sous-espace invariant associé coûte $O(n^3)$ opérations ou $O(n^2)$ si $m \ll n$. Une procédure Lapack estime ces conditionnements : c'est xTRSEN.

Remarque: Si la matrice est symétrique, on calcule ces quantités "à la main", puisque $\|P\|_2 = 1$ et $sep\,(T_{11}, T_{22}) = dist\,(sp\,(T_{11}), sp\,(T_{22}))$

2.4 La stabilité des algorithmes

La stabilité d'un algorithme permet de déterminer l'influence des erreurs d'arrondi sur le résultat [9].
Considérons le problème suivant : Calculer x tel que $F(x,d)=0$, l'idée de l'analyse régressive [26] est de montrer que la solution approchée $\overline{x}$ est la solution exacte d'un problème approché, c'est-à-dire $F(\overline{x},\overline{d})=0$ avec $\|\overline{d}-d\|$ aussi petit que possible. On suppose que le système flottant est consistant c'est-à-dire que pour toute opération arithmétique ou fonction élémentaire T et pour toute opération ou fonction g il existe une constante K telle que :

$$\forall x,y \in \mathcal{F} \qquad |fl(xTy)-xTy| \leq K\epsilon$$

$$\forall x \in \mathcal{F} \qquad |fl(g(x))-g(x)| \leq K\epsilon$$

où $\mathcal{F}$ est un ensemble de flottants et où ϵ est défini par :

$$\forall x \in \mathbb{R} \text{ tel que } fl(x) \text{ existe } |fl(x)-x| \leq |x|\epsilon$$

Définition : [5] Un algorithme est numériquement stable dans $\mathcal{V}$ si pour tout $d \in \mathcal{V}$ tel que $F(x,d)=0$, l'ensemble $\overline{D}(d)=\{\overline{d} \text{ tel que } F(\overline{x},\overline{d})=0\}$ n'est pas vide et s'il existe une constante K telle que

$$\forall d \in \mathcal{V} \qquad \inf_{\overline{d}\in\overline{D}(d)} \|\overline{d}-d\| \leq K\epsilon$$

L'erreur retour d'un algorithme stable dans $\mathcal{V}$ est donnée par :

$$B\epsilon = \sup_{d\in\mathcal{V}} \inf_{\overline{d}\in\overline{D}(d)} \|\overline{d}-d\|$$

Remarque : Soit Δd une perturbation des données, si l'on considére les erreurs d'arrondi, on obtient l'estimation d'erreur suivante :

$$\|\Delta\,\overline{x}\| \leq C\,(B\epsilon + \|\Delta d\|)$$

où $\Delta\,\overline{x} = fl\,(x+\Delta x)-x$, et C est le conditionnement du problème : calculer x tel que $F(x,d)=0$.

2.4.1 Cas des procédures Lapack

On peut montrer les résultats suivants [27, 11] :

1. **Mettre sous forme de Hessenberg**
   La matrice H calculée vérifie $H = Q^*\,(A+E)\,Q$, avec $\|E\|_F \leq c\,n^2\,\epsilon\,\|A\|_F$ où c est une petite constante et n est l'ordre de la matrice A.

2. **Méthode de tridiagonalisation de Householder**
   Soit A la matrice symétrique d'ordre n à tridiagonaliser et $\lambda_i,\ i = 1, n$ ses valeurs propres. Soit T la matrice tridiagonale calculée par la méthode de Householder et $\mu_i,\ i = 1, n$ ses valeurs propres. On montre alors :
   $$\sqrt{\frac{\sum(\mu_i - \lambda_i)^2}{\sum \lambda_i^2}} \leq 40\,n\,\epsilon\,(1 + 20\,\epsilon)^{2n}$$

3. **Méthode QR**
   Soit T la matrice de Schur calculée par cette méthode, elle vérifie $T = Q^*\,(A + E)\,Q$ avec $\|E\|_2 \simeq \epsilon\,\|A\|_2$

4. **Méthode de bisection**
   Soit T une matrice tridiagonale symétrique d'ordre n d'éléments diagonaux $\alpha_i,\ i = 1, n$ et sub-diagonaux $\beta_i,\ i = 1, n$.
   Cette méthode consiste à approcher une valeur propre de T en évaluant successivement les suites de Sturm en une valeur μ qui tend vers λ.
   On peut montrer que pour tout μ les valeurs calculées de la suite de Sturm : $p_0(\mu), ..., p_n(\mu)$ sont les valeurs exactes d'une matrice tridiagonale d'éléments diagonaux $\alpha_i + \delta\alpha_i,\ i = 1, n$ et sub-diagonaux $\beta_i + \delta\beta_i,\ i = 1, n$, où $\delta\alpha_i$ et $\delta\beta_i$ vérifient pour $i = 1, n$:
   $$|\delta\alpha_i| \leq (3.01)\,\epsilon\,(|\alpha_i| + |\mu|)$$
   $$|\delta\beta_i| \leq (1.51)\,\epsilon\,(|\beta_i|)$$

5. **Méthode des itérations inverses**
   Le vecteur propre x calculé est tel que (λ, x) soit un élément propre d'une matrice $A + E$ avec $\|E\|_\infty \leq c\,\epsilon\,\|A\|_\infty$ où c est une constante de l'ordre de l'unité.

   On peut donc considérer dans ce qui suit que toutes les procédures LAPACK sont stables.

2.5 Estimation d'erreur

On désire valider un résultat de calcul de valeurs et/ou vecteurs propres c'est-à-dire que l'usager utilise un enchaînement de procédures de calcul de valeurs et/ou vecteurs propres et veut avoir une **estimation** de l'erreur commise sur le résultat qu'il obtient.
On vient de voir que la qualité de ce résultat dépend du conditionnement de la valeur ou du vecteur propre calculé et de la stabilité de l'algorithme utilisé. N'oublions pas non plus que le résultat dépend de la précision des données.

Pour estimer l'erreur commise sur le résultat il faudra donc en théorie :

1. Evaluer l'erreur commise sur la matrice (ΔA)
2. Déterminer la stabilité arithmétique des algorithmes utilisés et estimer l'erreur régressive (B)
3. Estimer le conditionnement de la valeur ou du vecteur propre calculé si c'est nécessaire (C)

Le système utilise la formulation d'erreur suivante : $C * (B\epsilon + \Delta A)$, où ϵ représente la précision de la machine utilisée.

Dans la mesure où les procédures utilisées sont stables et où il est très difficile d'estimer B, on choisit une valeur arbitraire $B = 1$ pour tous les calculs, ce qui revient à tenir compte essentiellement de l'erreur commise sur la matrice.
En toute rigueur, il faudrait estimer l'erreur ΔH commise lors de la transformation de la matrice et utiliser cette borne dans la suite. Puisque cette opération est bien conditionnée, on suppose pour simplifier que $\Delta H = \Delta A$.
On a donc les estimations d'erreur simplifiées suivantes :

$$|\Delta\lambda| \leq C_\lambda \, (\epsilon + \|\Delta A\|)$$

$$|\theta_{max}(M, M')| \leq C_\theta \, (\epsilon + \|\Delta A\|)$$

où C_λ et C_θ représentent respectivement les estimations des conditionnements des valeurs et vecteurs propres, et où $\Delta\lambda$ et $\theta_{max}(M, M')$ représentent

respectivement l'erreur faite lors du calcul d'une valeur propre et d'un sous espace invariant.

L'estimation des conditionnements C_λ et C_θ dépend de la matrice :

- si la matrice est symétrique, le conditionnement d'une valeur propre simple ou multiple est toujours égal à 1 et le conditionnement d'un sous espace invariant de dimension m est égal à la distance du bloc de valeurs propres associées au reste du spectre. Le choix de m dépend des résultats ; en pratique, il faut grouper des valeurs propres voisines.
- si la matrice est non symétrique, les conditionnements C_λ et C_θ peuvent être estimés par une routine expert lors du calcul des valeurs et vecteurs propres ou après calcul par une procédure de Lapack. Là encore, le choix de la dimension m dépend des résultats.

Deuxième partie
Maquette

3 Logiciel utilisé

Pour réaliser le système, on a utilisé le logiciel SHIRKA [21] qui utilise une représentation par objets des connaissances.
Un **objet**, entité statique, est un assemblage structuré de toutes les caractéristiques se rapportant à un **concept**. La spécification de cet assemblage se fait par des **attributs**.
L'ensemble des objets d'un domaine est organisé dans une **structure hiérarchique** formant un treillis de **classes** qui se transmettent des informations par le mécanisme d'**héritage**. La dynamique des objets est due à la **classification**.
Une classe est un moule à partir duquel on fabrique des exemplaires, appelés **instances** de cette classe.
L'héritage est le mécanisme par lequel, lors de la classification, une classe transmet des informations à ses sous-classes ou à ses instances. Les informations transmises sont les attributs.
L'héritage peut être simple ou multiple. Dans le cas d'un héritage multiple, une sous-classe hérite de plusieurs classes, c'est-à-dire qu'elle hérite de l'union des informations contenues dans ces classes.
La classification permet de situer l'instance dans la hiérarchie en fonction des valeurs de ses différents attributs et d'en déterminer d'autres par **inférence**.

Les objets vont servir à représenter des contextes mathématiques.
Les attributs représenteront des propriétés mathématiques : les informations sur les propriétés de la matrice, sur le nombre d'éléments propres à calculer, sur la précision des calculs demandée...
Une hiérarchie de classes sera créée grâce aux différentes valeurs que peuvent prendre ces attributs.
Dans certaines classes, on aura assez d'informations sur le problème à résoudre pour pouvoir déterminer la procédure de résolution de celui-ci. "Un" attribut contiendra le nom de cette procédure.

Il est parfois intéressant de regrouper certains attributs. Pour cela, on définit des classes d'attributs. On pourra, par exemple, regrouper les attributs contenant les noms des procédures. Ainsi, pour déterminer une procédure,

on lancera la classification sans prendre en compte cette classe d'attributs, puisque le but est ici de déterminer les attributs de cette classe, ce qui se fera grâce à l'inférence. Par contre, pour valider une procédure il faudra lancer la classification en prenant en compte, entre autres, les attributs de cette classe. De plus, dans ce cas, aucun attribut n'est à déterminer, la classification sera donc lancée sans inférence. Ces classes d'attributs permettent de ne créer qu'une seule base, valable à la fois pour le choix et la validation de procédure.

4 Base de connaissances

L'usager fournit au système des informations sur une matrice (réelle ou complexe, symétrique ou non, stockage), sur le nombre de valeurs et vecteurs propres à calculer et sur la précision de ces calculs. Puis il peut choisir entre :

- Déterminer la(les) procédure(s) Lapack à utiliser pour calculer les valeurs et/ou vecteurs propres de cette matrice.
- Valider un enchainement de procédures (qu'il indique) pour le calcul de ces valeurs et/ou vecteurs propres.
- Valider le résultat d'une procédure (qu'il a exécutée hors-système) en calculant le conditionnement si c'est nécessaire, et choisir une autre procédure si le résultat est mauvais.

Les différents concepts de la base de connaissance se décomposent en trois groupes :

- Les concepts *Matrice, Pb, Type* permettant de définir le problème (ce sont les données) :
  - **Matrice** : définit la matrice grâce à trois attributs : **réel, symétrique, stockage** ;
    - **Réel**, booléen, est vrai si la matrice est réelle, faux si elle est complexe.
    - **Symétrique**, booléen, est vrai si la matrice est symétrique, faux si elle ne l'est pas.
    - **Stockage**, chaine de caractères, indique le stockage de la matrice : Dense, Tableau, Bande, Tridiagonal.
  - **Pb** : définit le nombre de valeurs et vecteurs propres désirés grâce à deux attributs **nb-valp** et **nb-vectp** de type chaine de caractères dont les valeurs admissibles sont : Zéro, Un peu, Beaucoup.

- **Type** : définit le type de calculs à faire : réels ou complexes, en simple ou en double précision. Les deux premiers attributs sont **mat** qui est la matrice définie précédemment et **précision**, chaine de caractères, dont les valeurs admissibles sont : simple ou double. Le troisième attribut **premlet**, chaîne de caractères, (classe d'attributs : **rés**) est déterminé lors de la classification et permet de connaître la première lettre des procédures Lapack à utiliser.

- Les concepts *Driver-routine, Transfo-matrice, Calc-val-prop, Calc-vect-prop* permettant de déterminer ou de valider les procédures à utiliser pour résoudre un problème (attribut **problème** (concept *Pb*)) faisant intervenir la matrice définie par l'attribut **mat** (concept *Matrice*) :

  Pour chacun de ces quatre concepts, un attribut de type chaîne de caractères, (respectivement *driver, mét-mat, mét-valp, mét-vectp*) est réservé pour le nom de la procédure. Cet attribut (classe d'attributs : **rés**) est soit déterminé lors de la classification (détermination de procédure), soit fourni par l'utilisateur (validation de procédure).

  - **Driver-routine** : ses attributs sont **mat, problème, driver**(nom de la driver-routine). De plus, dans le cas d'une matrice non symétrique, on indique si l'on désire estimer les conditionnements relatifs aux calculs effectués (attribut **calcul-conditionnement**, booléen), et si l'on désire récupérer les vecteurs de Schur (attribut **vect-Schur**, booléen).
  - **Transfo-matrice** : attributs **mat, mét-mat**(nom de la procédure de réduction de la matrice). Un dernier attribut **état-mat** (classe d'attributs : **sortie**) permet de connaître, après classification, l'état de la matrice réduite (matrice de Hessenberg ou tridiagonale réelle symétrique).
  - **Calc-val-prop :** attributs **mat, problème, état-mat** et **mét-valp**(nom de la procédure de calcul des valeurs propres).
  - **Calc-vect-prop** : attributs **mat, problème, état-mat, mét-valp** et **mét-vectp**(nom de la procédure de calcul des vecteurs propres).

- Les concepts permettant de valider un résultat :

  - **Calc-cond** : détermine si le calcul des conditionnements des valeurs et/ou vecteurs propres est nécessaire et par quelle procédure

il doit être fait. En entrée on donne les attributs **mat** (concept matrice), **problème** (concept pb) et **cond-connu**, booléen. De plus, si la matrice est non symétrique on indique la valeur de l'attribut **état-mat**. Eventuellement, on indique si la valeur propre est considérée simple ou multiple via l'attribut **type-valp**, chaine de caractères. On connait alors les calculs à faire en récupérant, après classification, l'attribut **calcul** (classe d'attributs : **rés**) et le nom de la procédure à utiliser pour ces calculs via l'attribut **proc** (classe d'attributs : **rés**).

- **Précision** : estime l'erreur faite sur le résultat, à partir des valeurs des attributs de type réel **cond, stab, delta** et **epsilon**, en utilisant la formule : $cond * (stab * epsilon + delta)$. Les valeurs de *cond, delta, epsilon* seront demandées à l'usager, avec possibilité de ne pas répondre pour *delta*, tandis que *stab* sera fixée à 1 dans le système. L'estimation du résultat est à récupérer dans l'attribut **prec**.
- **Qualité-résultat** : fournit un diagnostic (attribut **diagnostic**, chaîne de caractères, classe d'attributs : **rés**) sur le résultat. Les attributs à connaître sont **satisfait**, booléen, et éventuellement **premlet**, chaîne de caractères.

5 Utilisation du système

5.1 Détermination de la procédure

Dans ce paragraphe, le problème est le suivant : l'usager a une matrice et désire en calculer des valeurs et/ou vecteurs propres.

On va, grâce aux concepts présentés précédemment, déterminer la procédure à utiliser pour le calcul de ces valeurs et/ou vecteurs propres et la communiquer à l'utilisateur.

On indique, ici, la démarche à suivre pour déterminer cette procédure. Toutes les classifications devront être lancées **avec inférence** et sans prendre en compte la classe d'attributs *rés*.

- Créer une instance du concept **matrice**, afin de recueillir les propriétés de la matrice qui seront nécessaires par la suite.

- Créer une instance du concept **pb**, afin de déterminer le nombre de valeurs et/ou vecteurs propres désirés.
- Créer une instance du concept **type** ; ceci définit le type de calculs à faire. Classer cette instance : on pourra ainsi récupérer l'attribut *premlet*.
- Créer et classer une instance du concept **driver-routine**, où les attributs *mat* et *problème* sont ceux qui ont été définis précédemment.

Deux cas peuvent alors se produire :

- On a une feuille de l'arbre qui est classée "sure" : ceci correspond avec le fait que l'attribut *driver* est valué. Cela signifie qu'il existe une driver-routine qui résout le problème posé. Il suffit alors de récupérer la valeur de l'attribut *driver*, puisque c'est celle-ci qu'il faudra afficher.
- Aucune feuille n'a été classée "sure" : il n'existe donc pas de driver-routine qui résout ce problème. Il faudra donc créer et classer successivement :
  - Une instance du concept **transfo-matrice**. En plus de la classe d'attributs *rés*, ne pas considérer la classe *sortie* lors de la classification. Une fois l'instance classée, récupérer la valeur de l'attribut *état-mat*, afin de le réinjecter comme attribut de l'instance suivante.
  - Une instance du concept **calc-val-prop**. Ici, aussi, on récupère la valeur de l'attribut *état-mat* qui peut avoir été modifié.
  - Une instance du concept **calc-vect-prop**.

  Pour chacune de ces trois instances, il doit y avoir, au plus, une feuille classée "sure". Visualiser les attributs de ces feuilles, afin de récupérer les valeurs éventuelles des attributs *mét-mat, mét-valp* et *mét-vectp*. Ce sont ces attributs qu'il faudra afficher.

Selon le cas, il se peut qu'il n'y ait qu'un nom de procédure à afficher (attribut *driver*) ou plusieurs (attributs *mét-mat, mét-valp* et *mét-vectp*). Or, ces attributs ne contiennent que la fin du nom des procédures. En effet, la première lettre de la procédure a été déterminé par l'attribut *premlet*. Donc, pour chaque procédure, il faudra commencer par afficher sa première lettre grâce à l'attribut *premlet*, puis le reste du nom de la procédure qui

sera contenu, suivant le cas, dans l'attribut *driver, mét-mat, mét-valp* ou *mét-vectp*.

Il existe cependant un problème pour l'affichage des procédures xSTEBZ et xSTERF qui sont des procédures n'utilisant que des nombres réels. Il n'y a donc pas de procédure spécifique au cas complexe. Ainsi, CSTEBZ et ZSTEBZ n'existent pas, il faudra donc conseiller respectivement les procédures SSTEBZ et DSTEBZ. De même CSTERF et ZSTERF deviennent respectivement SSTERF et DSTERF.

5.2 Validation de procédure

Le but est, ici, de déterminer si un enchaînement de procédures proposé par l'usager résout effectivement le problème aux valeurs propres donné. Donc, en plus de la matrice et du problème, l'utilisateur spécifie une liste de procédures.

Pour réaliser ceci, il faut procéder de la manière suivante :

- Créer des instances des concepts **matrice** et **pb**.
- Créer et classer (**avec inférence** et sans prendre en compte la classe d'attributs *rés*) une instance du concept **type**, où les attributs *mat* et *problème* seront donnés en entrée.
- Créer et classer (**avec inférence** et sans prendre en compte la classe d'attributs *rés*) quatre instances du concept **nom-routines**, où *typ* (concept type) sera donné en entrée, ainsi que l'attribut *arbre*. On récupère la liste des procédures existantes pour l'arbre considéré (attribut *arbre*) et le type des calculs demandé (attribut *typ*), dans l'attribut *proc* (classe d'attributs *rés*). On crée quatre instances en vue de déterminer :
  1. les driver-routines (*arbre* = "driver-routine")
  2. les procédures de transformations de matrice (*arbre* = "transfo-matrice")
  3. les procédures de calcul de valeurs propres (*arbre* = "calc-val-prop")

4. les procédures de calcul de vecteurs propres (*arbre* = "calc-vect-prop")

Contrairement à la détermination de procédure, toutes les classifications suivantes sont à faire **sans inférence** et en prenant en compte la classe d'attributs *rés*.

- Si l'usager dispose d'une driver-routine : créer et classer une instance du concept **driver-routine**, où on valuera aussi l'attribut *driver*.

  Si une feuille est classée "sure", on peut résoudre le problème posé avec la procédure proposée qui est une driver-routine : le choix de procédure est correct.
  Sinon, l'utilisateur ne peut pas résoudre son problème avec cette driver-routine : son choix est mauvais.
  Dans les deux cas, il est inutile de passer à l'étape suivante, puisqu'on peut répondre à l'utilisateur.

- Si l'usager dispose d'une procédure de transformation de matrice (sinon on passe à l'étape suivante ; en effet, il se peut que l'usager dispose d'une matrice déjà sous forme réduite et peut donc directement calculer les valeurs et/ou vecteurs propres) : créer et classer une instance du concept **transfo-matrice**, où il faudra aussi valuer l'attribut *mét-mat*. On ne valuera pas *état-mat* qui sera déterminé lors de la classification.

  Si une feuille est classée "sure", on peut réduire la matrice avec la procédure proposée : on passe à l'étape suivante.
  Sinon, cette procédure ne résout pas le problème posé. Il est inutile de poursuivre, puisque le choix de procédure est mauvais.

- Si l'usager dispose d'une procédure de calcul de valeurs propres (sinon on passe à l'étape suivante ; en effet, il se peut que l'usager désire seulement calculer des vecteurs propres s'il connait déjà les valeurs propres) : créer et classer une instance du concept **calc-val-prop**, où il faudra aussi valuer l'attribut *mét-valp*.

  Si une feuille est classée "sure", on peut calculer les valeurs propres avec la procédure proposée : on passe à l'étape suivante.
  Sinon, le choix de procédure est mauvais : inutile de continuer.

- Si l'usager dispose d'une procédure de calcul de vecteurs propres : créer et classer une instance du concept **calc-vect-prop**, où il faudra aussi valuer l'attribut *mét-vectp*.

Si une feuille est classée "sure", on peut calculer les vecteurs propres avec la procédure proposée. Le choix de procédure est correct.
Sinon, cette procédure n'est pas acceptable pour ce problème.

5.3 Validation de résultat

On s'intéresse, ici, au problème suivant : un calcul de valeurs et/ou vecteurs propres a été réalisé grâce à une ou plusieurs procédures Lapack et l'utilisateur désire savoir s'il peut ou non faire confiance au résultat obtenu (voir paragraphe 2.5).

On procédera de la manière suivante :

- Afin de savoir si le calcul des conditionnements est à faire ou non et, si oui, par quelle procédure le faire, on va :
  - Créer une instance du concept **calc-cond** sans valuer les attributs de la classe *rés*.
  - Classer cette instance (classification **avec inférence** et sans prendre en compte les attributs de la classe *rés*).
  - Visualiser les attributs *calcul* et *proc*
- pour savoir si l'utilisateur est satisfait du résultat, il faut :
  - Créer une instance du concept **précision** sans valuer l'attribut *prec*.
  - Classer cette instance (classification avec inférence).
  - Visualiser l'attribut *prec*.
  - Créer une instance du concept **qualité-résultat** sans valuer l'attribut de la classe *rés*.
  - Classer cette instance (classification avec inférence et sans prendre en compte l'attribut de la classe *rés*).
  - Visualiser l'attribut *diagnostic*.

Conclusion

L'avantage d'un tel système est qu'il est possible d'agrandir la base assez facilement et sans toucher à ce qui a été fait. En effet, ce qui a été fait correspond à une catégorie de problèmes existants actuellement. Si, à l'avenir, de nouveaux problèmes apparaissaient, il suffirait de greffer ces nouveaux cas sur ce qui existe. Les greffes pourraient consister à élargir le domaine de certains attributs et à créer des sous classes correspondant à ces nouvelles valeurs. Par exemple, le cas des grandes matrices creuses est absent du système car il n'est pas, pour l'instant, traité dans Lapack. Cependant, il y sera intégré dans le futur et il faudra l'insérer dans le système expert. Pour ceci, il suffira d'élargir le domaine de l'attribut *stockage* du concept *matrice* afin de créer une sous-classe *matrice-creuse*, et dans les concepts de détermination de procédures on ajoutera des sous-classes relatives aux matrices creuses.

Cette modélisation par objets a permis de n'utiliser qu'une seule base valable à la fois pour la détermination et la validation de procédure. Ainsi, les connaissances ne sont pas dupliquées inutilement. De plus, la modélisation par objets est naturelle car chaque objet représente un contexte mathématique.

Cependant, en l'état actuel, l'utilisation du système reste collée à l'intelligence artificielle et nécessite quelques connaissances dans ce domaine. C'est pourquoi, une interface est actuellement en cours de développement à l'IRIMAG : elle permettra l'utilisation du système par tout numéricien. Aussi, dans le cas d'un échec du système pour valider une procédure, l'intelligence artificielle permet d'expliquer la cause de cet échec, mais les explications restent en termes d'objets et de classes et sont peu intelligibles pour un numéricien qui ne connaît pas le système. Il serait intéressant que ce coté intelligence artificiel soit transparent pour l'utilisateur.

On a décidé de ne pas générer les séquences d'appel aux procédures sélectionnées, car ceci aurait énormément augmenté la complexité des bases. En effet, il faut, par exemple, traiter les cas particuliers où l'utilisateur ne désire calculer que des valeurs ou vecteurs propres. De plus, il faut prendre en compte le fait que l'enchaînement de ces procédures fait intervenir des procédures intermédiaires sans intérêt du point de vue mathématique. Ces procédures intermédiaires sont actuellement conseillées dans des commentaires.

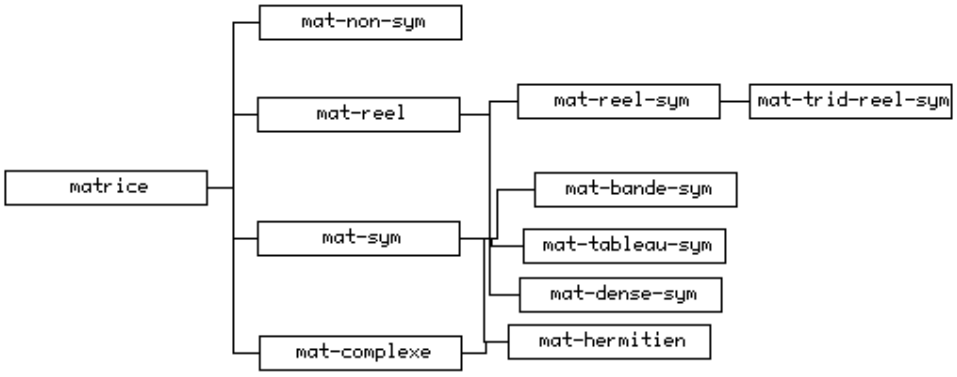
L'analyse d'échec repose ici sur la stabilité numérique des algorithmes utilisés et sur l'estimation de conditionnement fourni par LAPACK. Le cas des valeurs propres multiples est traité par le choix d'une taille de bloc qui est laissé à l'usager. Il serait intéressant d'approfondir l'analyse dans ce cas, en particulier pour des valeurs propres défectives, mais l'expertise numérique dans ce domaine est trop incomplète pour l'intégrer dans un système expert.

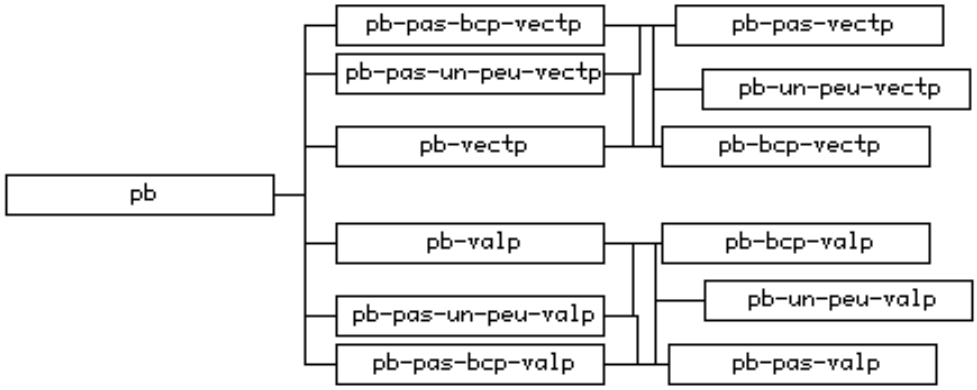
Annexes

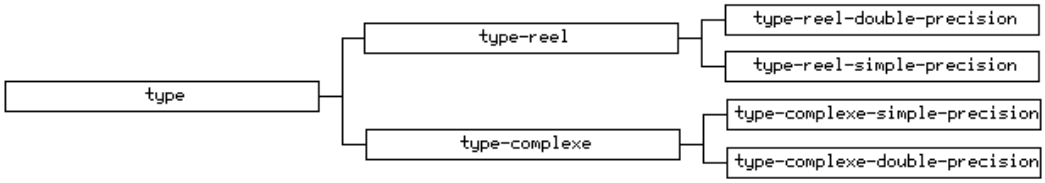
On présente, ci-après, successivement les graphes des concepts suivants :

1. Matrice, Pb, Type
2. Driver-routine
3. Transfo-matrice, Calc-val-prop, Calc-vect-prop
4. Nom-routines
5. Calc-cond, Précision, Qualité-résultat

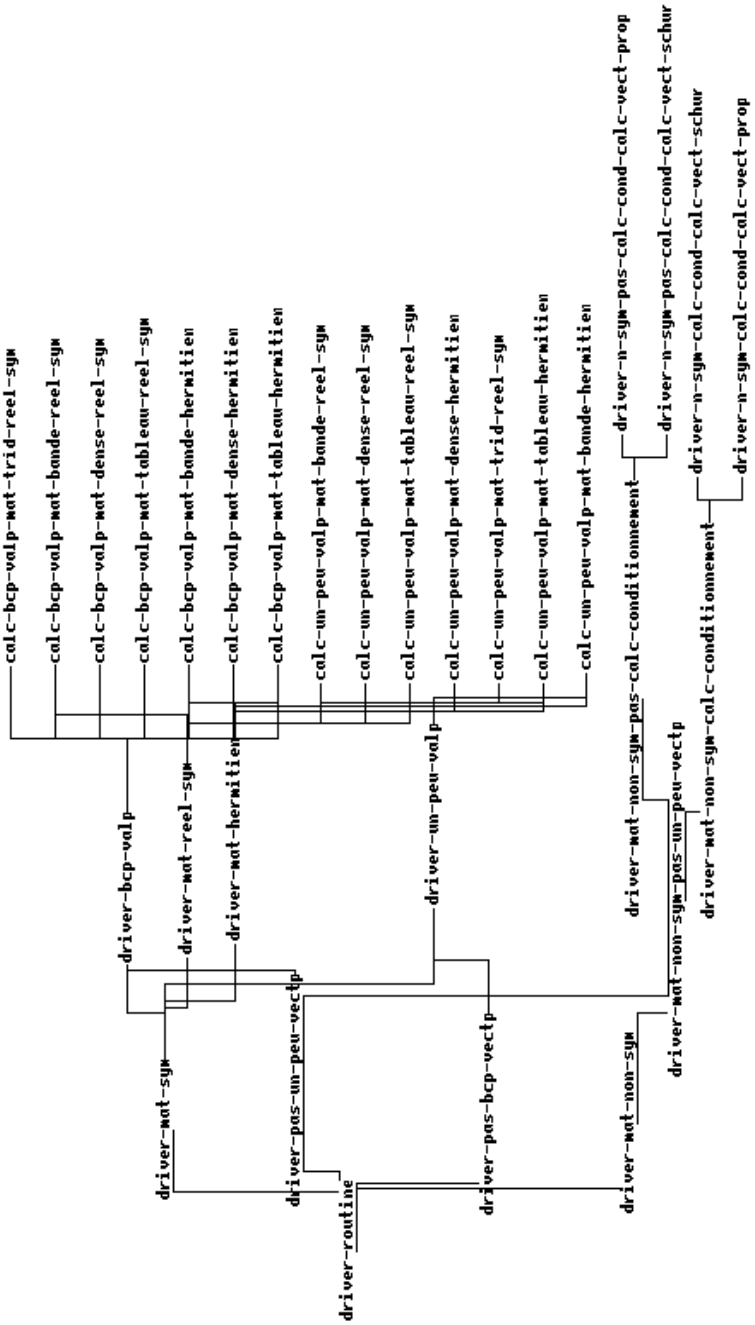
Concepts *Matrice, Pb, Type*



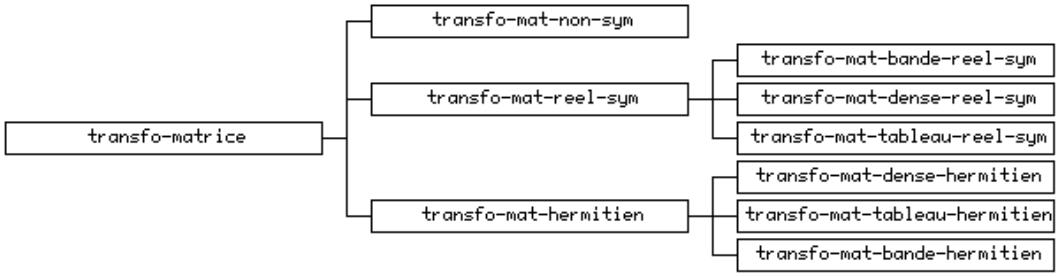


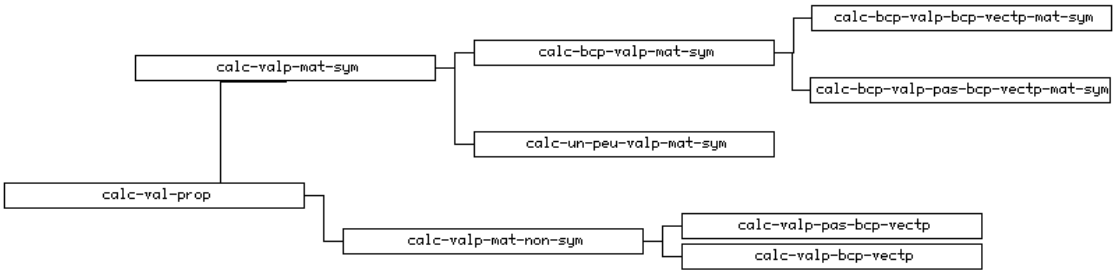


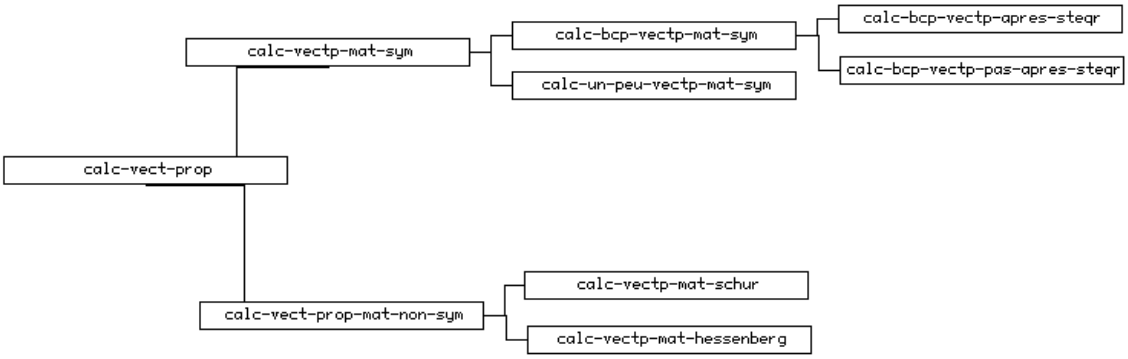
Concept *Driver-routine*



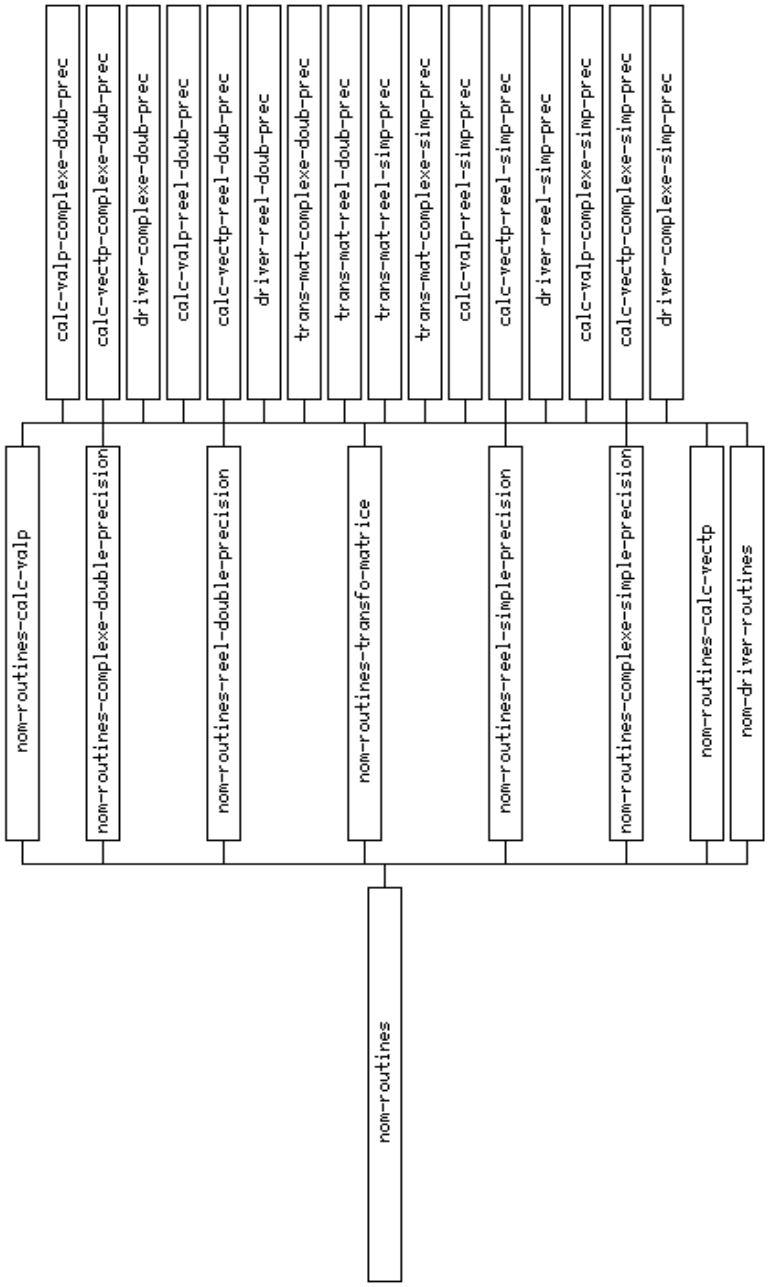
Concepts *Transfo-matrice, Calc-val-prop, Calc-vect-prop*



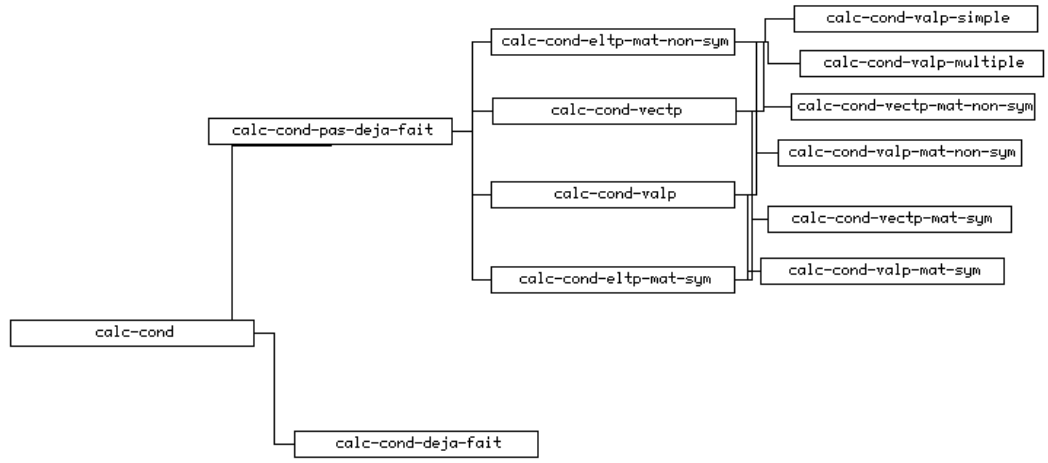


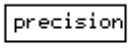


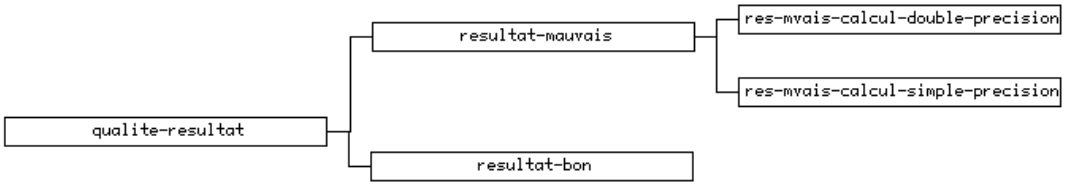
Concept *Nom-routines*



Concepts *Calc-cond, Précision, Qualité-résultat*







Bibliographie

[1] E. Anderson and al. *LAPACK Users' guide* SIAM, 1992.

[2] Z. Bai, J. Demmel, A. McKenney *On the conditioning of the nonsymmetric eigenproblem : Theory and software.* LAPACK Working Note 13, 1989.

[3] F. Bauer, C. Fike. *Norms and exclusion theorems.* Num. Math. 2, 1960.

[4] F. Chatelin. *Valeurs propres de matrices.* Collection mathématiques appliquées pour la maîtrise, Masson, 1988.

[5] F. Chatelin, V. Frayssé. *Arithmetic reliability of Algorithms.* Symposium on High Performance Computers, Montpellier, Octobre 1991.

[6] P. Ciarlet. *Introduction à l'analyse matricielle.* Collection mathématiques appliquées pour la maîtrise, Masson, 1984.

[7] J.W. Demmel. *Computing stable eigendecomposition of matrices.* Linear algebra and applications, No 79, 1986.

[8] J. Erhel and B. Philippe, *Design of a toolbox to control arithmetic reliability*, in SCAN'91, Oldenburg, 1991.

[9] J. Erhel. *Statistical estimation of roundoff errors and condition numbers.* Rapport de recherche INRIA No 1490, 1991.

[10] *FOCUS Consortium, FOCUS : Internal reports of the FOCUS project.* Contact NAG Limited, Oxford UK (1988/1989).

[11] G.H. Golub, C.F. Van Loan. *Matrix computations.* Johns Hopkins University Press, 1983.

[12] B. Kagstrom, A. Ruhe. *An algorithm for numerical computation of the Jordan normal form of a complex matrix.* ACM Trans. Math. Software 6 :398-419, 1980.

[13] C.M. O'Brien. *The GLIMPSE System.* NAG Technical Report TR 12/88. NAG Limited, Oxford, UK (September 1988).

[14] W. Kahan, B.N. Parlett, E. Jiang. *Residual bounds on approximate eigensystems of nonnormal matrices.* SIAM J.Numer.Anal, Vol 19, 1982.

[15] *KASTLE: Internal Project Reports of The Teaching Company Scheme Project.* NAG Limited, Oxford, UK (1987/1988).

[16] T. Kato. *Perturbation theory of linear operators.* Springer-Verlag, Berlin, 1966.

[17] S. Konig *An expert system shell for validated computation and its application to solving linear equations.* In SCAN 91, Oldenburg, 1991.

[18] P. Laug. *Pilotage d'un code modulaire d'éléments finis par un système expert.* Rapport de recherche 653, INRIA, Mars 1987.

[19] G. Masini, A. Napoli *Les Langages a objets: langages de classes, langages de frames, langages d'acteurs.* InterEditions, 1989.

[20] B.N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice Hall, Englewood Cliffs, 1980.

[21] F. Rechenmann, P. Fontanille, P. Uvietta. *SHIRKA : Système de gestion de bases de connaissances centrées objets: manuel d'utilisation.* INRIA et laboratoire ARTEMIS/IMAG, 1990.

[22] F. Rechenmann and B. Rousseau, *A development shell for knowledge-based systems in scientific computing*, in Houstis E.N., Rice J.R. (éds), Expert Systems for Numerical Computing, Elsevier Science Publishers, Amsterdam, 1992.

[23] C. Saurel. *Contribution aux systèmes experts: développement d'un cas concret et étude du problème de la génération d'explications négatives.* Thèse d'informatique, Toulouse, 15 décembre 1987.

[24] G.W. Stewart. *Error and perturbations bounds for subspaces associated with certain eigenvalue problems.* SIAM Review, Vol 15, No 4, 1973.

[25] J.M. Varah. *On the separation of two matrices.* SIAM J.Numer.Anal, Vol 16, No 2, 1979.

[26] J.H. Wilkinson. *Rounding errors in algebraic processes.* Englewoods Cliffs, N.J.: Prentice-Hall, 1963.

[27] J.H. Wilkinson. *The algebraic eigenvalue problem.* Oxford University Press, 1965.

Unité de recherche INRIA Lorraine, Technôpole de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105,
78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS
Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
(France)
ISSN 0249-6399